**Application Hardening**

The first step for any organization should be to create a detailed (and accurate) application inventory of both internal and external-facing applications. This can be achieved in many ways, and blue teams on a budget could benefit from pentesting tools such as Nmap and EyeWitness to assist in the process. Various open-source and paid tools can be used to create and maintain this inventory. Without knowing what exists in the environment, we won't know what to protect! Creating this inventory may expose instances of "shadow IT" (or unauthorized installs), deprecated applications that are no longer needed, or even issues such as a trial version of a tool being converted to a free version automatically (such as Splunk when it no longer requires authentication).

**General Hardening Tips**

The applications discussed in this section should be hardened to prevent compromise using these techniques and others. Below are some important measures that can help secure deployments of WordPress, Drupal, Joomla, Tomcat, Jenkins, osTicket, GitLab, PRTG Network Monitor, and Splunk in any environment.

* Secure authentication: Applications should enforce strong passwords during registration and setup, and default administrative account passwords should be changed. If possible, the default administrative accounts should be disabled, with new custom administrative accounts created. Some applications inherently support 2FA authentication, which should be made mandatory for at least administrator-level users.
* Access controls: Proper access control mechanisms should be implemented per application. For example, login pages should not be accessible from the external network unless there is a valid business reason for this access. Similarly, file and folder permissions can be configured to deny uploads or application deployments.
* Disable unsafe features: Features such as PHP code editing in WordPress can be disabled to prevent code execution if the server is compromised.
* Regular updates: Applications should be updated regularly, and patches supplied by vendors should be applied as soon as possible.
* Backups: System administrators should always configure website and database backups, allowing the application to be quickly restored in case of a compromise.
* Security monitoring: There are various tools and plugins that can be used to monitor the status and various security-related issues for our applications. Another option is a Web Application Firewall (WAF). While not a silver bullet, a WAF can help add an extra layer of protection provided all the measures above have already been taken.
* LDAP integration with Active Directory: Integrating applications with Active Directory single sign-on can increase ease of access, provide more auditing functionality (especially if synced with Azure), and make managing credentials and service accounts more streamlined. It also decreases the number of accounts and passwords that a user will have to remember and give fine-grained control over the password policy.

Every application that we discussed in this module (and beyond) should be following key hardening guidelines such as enabling multi-factor authentication for admins and users wherever possible, changing default admin user account names, limiting the number of admins, and how admins can access the site (i.e., not from the open internet), enforce the principle of least privilege throughout the application, perform regular updates to address security vulnerabilities, taking regular backups to a secondary location to be able to recover quickly in the event of an attack and implement security monitoring tools that can detect and block malicious activity and account brute-forcing, among other attacks.

Finally, we should be careful with what we expose to the internet. Does that GitLab repo really need to be public? Does our ticketing system need to be accessible outside the internal network? With these controls in place, we will have a solid baseline to apply to all applications regardless of their function.

We should also perform regular checks and updates to our application inventory to ensure that we are not exposing applications on the internal or external network that are no longer needed or have severe security flaws. Finally, perform regular assessments to look for security vulnerabilities and misconfigurations as well as sensitive data exposure. Follow through on remediation recommendations included in your penetration testing reports and periodically check for the same types of flaws discovered by your penetration testers. Some could be process-related, requiring a mindset shift for the organization to become more security conscious.

**Application-Specific Hardening Tips**

Though the general concepts for application hardening apply to all applications that we discussed in this module and will encounter in the real world, we can take some more specific measures. Here are a few:

| **Application** | **Hardening Category** | **Discussion** |
| --- | --- | --- |
| [WordPress](https://wordpress.org/support/article/hardening-wordpress/) | Security monitoring | Use a security plugin such as [WordFence](https://www.wordfence.com/) which includes security monitoring, blocking of suspicious activity, country blocking, two-factor authentication, and more |
| [Joomla](https://docs.joomla.org/Security_Checklist/Joomla!_Setup) | Access controls | A plugin such as [AdminExile](https://extensions.joomla.org/extension/adminexile/) can be used to require a secret key to log in to the Joomla admin page such as http://joomla.inlanefreight.local/administrator?thisismysecretkey |
| [Drupal](https://www.drupal.org/docs/security-in-drupal) | Access controls | Disable, hide, or move the [admin login page](https://www.drupal.org/docs/7/managing-users/hide-user-login) |
| [Tomcat](https://tomcat.apache.org/tomcat-9.0-doc/security-howto.html) | Access controls | Limit access to the Tomcat Manager and Host-Manager applications to only localhost. If these must be exposed externally, enforce IP whitelisting and set a very strong password and non-standard username. |
| [Jenkins](https://www.jenkins.io/doc/book/security/securing-jenkins/) | Access controls | Configure permissions using the [Matrix Authorization Strategy plugin](https://plugins.jenkins.io/matrix-auth) |
| [Splunk](https://docs.splunk.com/Documentation/Splunk/8.2.2/Security/Hardeningstandards) | Regular updates | Make sure to change the default password and ensure that Splunk is properly licensed to enforce authentication |
| [PRTG Network Monitor](https://kb.paessler.com/en/topic/61108-what-security-features-does-prtg-include) | Secure authentication | Make sure to stay up-to-date and change the default PRTG password |
| osTicket | Access controls | Limit access from the internet if possible |
| [GitLab](https://about.gitlab.com/blog/2020/05/20/gitlab-instance-security-best-practices/) | Secure authentication | Enforce sign-up restrictions such as requiring admin approval for new sign-ups, configuring allowed and denied domains |

**Conclusion**

In this module, we covered a critical area of penetration testing: common applications. Web applications present an enormous attack surface and often go overlooked. During an external penetration test, often, the majority of our targets are applications. We must understand how to discover applications (and organize our scan data to process it efficiently), footprint versions, discover known vulnerabilities, and leverage built-in functionality. Many organizations do well with patching and vulnerability management but often overlook issues such as weak credentials to access Tomcat Manager or a printer with default credentials for the web management application where we can obtain LDAP credentials to use as a foothold into the internal network. The two skills assessments that follow are meant to put the application discovery and enumeration process to the test.